ELECTRONIC COMPUTER, SEMICONDUCTOR INTEGRATED CIRCUIT, CONTROL METHOD, PROGRAM GENERATION METHOD, AND PROGRAM

TECHNICAL FIELD

[0001]

The present invention relates to an electronic computer, semiconductor integrated circuit, control method, program generation method, and program, and a technology for executing a part or all of processing by an application program at high speed using reconfigurable hardware.

BACKGROUND ART

[0002]

In order to execute an application program beyond the processing capability of a CPU, the technique using special-purpose hardware has been conventionally proposed/developed. FIG. 30 shows a technique in which special-purpose hardware executes the whole application program. FIG. 31 shows a technique in which special-purpose hardware is prepared to execute a part of an application program having the hardware connected to a general-purpose CPU via a network, and a part of the application is executed by the hardware at high speed. FIG. 32 shows a technique in which a part of an application program is executed by special purpose hardware, and by adding a new instruction set that has the

special purpose hardware execute processing within a CPU, the part executed by the newly added instructions is processed at high speed. While the techniques described above in which all or a part of an application is implemented in hardware improve the processing capability a great deal, they require large costs because a new piece of hardware needs to be developed/manufactured for every application. Meanwhile, reconfigurable hardware where logic circuits such as FPGA (Field Programmable Gate Array) and PLD (Programmable Logic Device) can be created by a program has drawn some attention because it can execute particular processing specified by a program by changing the program and reconfiguring the logic circuit without changing devices. In recent years, methods for realizing an application program that requires high processing capability using reconfigurable hardware, instead of special purpose hardware, without having to newly manufacture hardware (i.e. low costs) and devices using these methods have been proposed. For example, reconfigurable hardware is used instead of special purpose hardware in Japanese Patent Kokai Publications No. JP 8-316329A and No. JP 11-184718A. Further, extended instructions added to a CPU are realized by reconfigurable hardware in JP 3099889B.

[0003]

[Patent Document 1]

Japanese Patent Kokai Publication No. JP-A-8-316329

[Patent Document 2]

Japanese Patent Kokai Publication No. JP-A-11-184718

[Patent Document 3]

Japanese Patent No. 3099889

[Patent Document 4]

Japanese Patent Kokai Publication No. JP-P2001-147802A

[Patent Document 5]

Japanese Patent Kohyo Publication No. JP-A-11-507478

DISCLOSURE OF THE INVENTION

PROBLEMS TO BE SOLVED BY THE INVENTION

[0004]

Since the above-mentioned conventional techniques do not consider the capacity of logic circuits created in reconfigurable hardware, an application program that exceeds the capacity of the reconfigurable hardware cannot be implemented. Therefore, the size of an application program is proportional to the size of the reconfigurable hardware in which the application program is implemented, resulting in high costs.

[0005]

Meanwhile, when an application program is implemented in reconfigurable hardware with small capacity in order to reduce costs, the application program needs to be divided considering the size of the logic circuit that can be implemented in the hardware resource. However, divided programs and the control between these programs greatly depend on the

architecture where the programs are implemented. Therefore, the divided programs and the control between these programs cannot be reused once the architecture has been changed, reducing design efficiency greatly.

[0006]

Further, the adequacy of the application program division (whether or not it can be implemented in the reconfigurable or reconstructable hardware) is not clear until the last stage of the reconfigurable hardware realization i.e. when the logic circuits are mapped onto the reconfigurable hardware, and if an improper division such as one exceeding the implementable size occurs, a great deal of redesigning will be necessary, reducing design efficiency considerably.

[0007]

Further, as described in Japanese Patent Kokai Publication No. JP-P2001-147802A and Japanese Patent Kohyo Publication No. JP-A-11-507478, when the control between divided processing operations is controlled by a CPU external to reconfigurable hardware, the control between divided processing operations is reusable as program data of the CPU, however, the divided processing operations are not since they depend on the architecture of the reconfigurable hardware, where the application program is implemented, and the implementable capacity. Further, since the control between the processing operations is performed by the CPU, a waiting time such as the

overhead of a system call occurs between the CPU and the reconfigurable hardware, resulting in performance degradation. The more the scale of the application program increases, the more serious these problems become.

[0008]

It is an object of the present invention to provide an electronic computer, control method, program generation method, and program wherein an application program is executed and is easily made reusable by dividing the application program into processing units, and by creating a logical circuit for every processing unit in reconfigurable (or reconstructable) hardware by switching so as to improve the processing speed at low cost.

MEANS TO SOLVE THE PROBLEM

[0009]

A first electronic computer of the present invention is characterized in that it comprises a processing device including reconfigurable (or reconstructable) hardware that can create a logic circuit with a program and a control device executing a command specified by the processing device, and the command is instructed to be executed when the processing device detects a predetermined condition and includes a command for execution of switching programs logically creating the reconfigurable hardware.

[0010]

A second electronic computer of the present invention in

the first electronic computer of the present invention is characterized in that the processing device comprises a plurality of banks (each) having a processing element with reconfigurable hardware and at least one program data memory holding a program that creates a logic circuit in the reconfigurable hardware, and an effective bank selection unit selecting one bank from the plurality of banks, making it effective and connecting it to the outside.

[0011]

A third electronic computer of the present invention in the first electronic computer of the present invention is characterized in that the processing device comprises a bank including a processing element that includes reconfigurable hardware, a plurality of program data memories (each) holding a program that creates a logic circuit in the reconfigurable hardware, and an effective block selection unit selecting one memory from the plurality of program data memories and making it effective.

[0012]

A fourth electronic computer of the present invention in the second or third electronic computer of the present invention is characterized in that at least one processing element of the processing device is comprised of reconfigurable hardware and the other processing elements are (each) comprised of reconfigurable hardware or a general-purpose CPU.

[0013]

A fifth electronic computer of the present invention in the second, third, or fourth electronic computer of present invention is characterized in that the control device interprets and executes an activate command specifying the effective bank in case where there is a plurality of the banks, and specifying the effective program data memory and activating operation of the specified processing element when there is a plurality of the program data memories; a halt command halting operation of the specified processing device; an interrupt command issuing an interrupt vector from the control device to the specified processing device; a load_prg command transferring program data from a specified memory device to the program data memory; a cancel_prg command canceling the load_prg instruction; and a wait_prg command waiting until completion of the load_prg instruction.

[0014]

A sixth electronic computer of the present invention in the first, second, third, fourth, or fifth electronic computer of present invention is characterized by comprising a command code memory holding commands that the control device executes wherein the control device comprises a command code reference device reading commands from the command code memory according to an address specified by the processing device, interpreting, and executing it.

[0015]

A seventh electronic computer of the present invention in the sixth electronic computer of present invention is characterized in that the command code reference device comprises an address counter holding the address of the command code memory, and in the exchange of commands between the processing device and the control device, a first address control line indicating that an address signal line outputted by the processing device is effective, and a second address counter control line instructing whether the value of the address signal line is stored in the address counter as it is or the result of adding the value of the address signal line to the value of the address counter is stored in the address counter when the first control line is effective.

[0016]

An eighth electronic computer of the present invention in the seventh electronic computer of the present invention is characterized in that the commands are stored in the command code memory in a format comprising a command code that classifies the commands, an address counter control code, and a flag that indicates whether or not the following command is executed, and the address counter control code includes a load_adr command setting the value of the address counter and a add_adr command adding a specified value to the address counter.

[0017]

A ninth electronic computer of the present invention in the eighth electronic computer of the present invention is characterized in that the address counter control code includes a push_adr command that hides (stores as a sidetruck) the address counter in an address counter stack provided in the control device and that sets a new value to the address counter, and a pop_adr command that returns the value of the address counter stack to the address counter.

[0018]

A tenth electronic computer of the present invention in any one of the first through ninth electronic computers of the present invention is characterized by comprising a cache device including a cache memory that temporarily holds data to be transferred to the processing device and a cache controller that controls the cache memory wherein the cache controller is controlled by a command issued by the processing device.

[0019]

An eleventh electronic computer of the present invention in the tenth electronic computer of the present invention is characterized in that the cache device comprises an address translation device that translates an address defined externally to the processing device into an address defined inside of the processing device, and the address translation device is controlled by a command issued by the processing device.

[0020]

A twelfth electronic computer of the present invention comprises a processing device including reconfigurable hardware that can create a logic circuit with a program, and a control device executing a command specified by the processing device; the command is instructed to be executed when the processing device detects a predetermined condition and includes a command that executes switching programs logically creating the reconfigurable hardware; and the processing device comprises a second processing device including reconfigurable hardware that can create a logic circuit with a program and a second control device executing a command specified by the second processing device.

[0021]

A semiconductor integrated circuit of the present invention implements any one of the first through eleventh electronic computers of the present invention.

[0022]

In a first control method of the present invention, when a processing device including reconfigurable hardware that can create a logic circuit with a program detects a predetermined condition, it issues an instruction to execute a command, and a control device that has received the command execution instruction from the processing device switches programs that logically creates reconfigurable hardware.

[0023]

A second control method of the present invention in the first control method of the present invention is characterized in that after the switching, while a program in a predetermined program data memory is being executed, a next program is read into another program data memory.

[0024]

In a third control method of the present invention, a processing device issues an instruction to execute a command when a predetermined condition is detected, wherein the processing device includes reconfigurable hardware, a plurality of program data memories that hold programs creating logic circuits of the reconfigurable hardware, and an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective; and

a control device that has received the command execution instruction from the processing device executes an activate command controlling the effective block selection unit so as to make the specified program data memory effective and connecting it to the reconfigurable hardware; and switching of the content of a logic circuit executed by the reconfigurable hardware is executed.

[0025]

A fourth control method of the present invention in the

second control method of the present invention is characterized in that the control device executes;

a halt command halting the operation of the specified processing device;

an interrupt command issuing an interrupt vector from the control device to the specified processing device;

a load_prg command transferring program data from a specified memory device to the program data memory;

a cancel_prg command canceling the load_prg instruction; and

a wait_prg command waiting until the completion of the load_prg instruction.

[0026]

A first program generation method of the present invention comprises; a control flow analysis procedure in which the control flow of an application program is analyzed, the application program is divided into processing units, and a command sequence intermediate code combining commands controlled by reconfigurable hardware that executes the divided processing units within an electronic computer is generated;

a command sequence implementation procedure in which a command sequences is generated by translating the command sequence intermediate code into a form that can be executed by the electronic computer; and

a program data generation procedure in which the

operational content of a processing unit is translated into a form that can be executed by the electronic computer.

[0027]

A second program generation method of the present invention in the first program generation method of the present invention is characterized in that the application program is divided so that each processing unit can be stored in a program data memory that holds a program creating a logic of the reconfigurable hardware when the control flow of the application program is analyzed and divided into processing units in the control flow analysis procedure.

[0028]

A first program of the present invention is characterized by having a computer execute a procedure in which, when a processing device including reconfigurable hardware that can create a logic circuit with a program detects a predetermined condition and issues an instruction to execute a command, a control device that has received the command execution instruction from the processing device executes switching programs logically creating the reconfigurable hardware.

[0029]

A second program of the present invention is characterized by having a computer execute a procedure in which, when a processing device including reconfigurable hardware, a plurality of program data memories that hold programs creating logic

circuits of the reconfigurable hardware, and an effective block selection unit that selects one program data memory from the plurality of program data memories and that makes it effective detects a predetermined condition and issues an instruction to execute a command, a control device that has received the command execution instruction from the processing device executes an activate command that controls the effective block selection unit so as to make the specified program data memory effective and that switches connection to the reconfigurable hardware.

[0030]

A third program of the present invention in the second program of the present invention is characterized by having a computer execute a procedure in which a halt command halting operation of the specified processing device, an interrupt command issuing an interrupt vector from the control device to the specified processing device, a load_prg command transferring program data from a specified memory device to the program data memory, a cancel_prg command canceling the load_prg instruction, and a wait_prg command waiting until the completion of the load_prg instruction are executed.

BRIEF DESCRIPTION OF THE DRAWINGS

[0031]

FIG. 1 is a block diagram illustrating the structure of an embodiment of the present invention.

FIG. 2 is a block diagram showing an example of a processing device of the embodiment of the present invention.

FIG. 3 is a diagram showing an example of a processing element of the embodiment of the present invention.

FIG. 4 is a diagram showing an example of a processing element of the embodiment of the present invention.

FIG. 5 is a diagram showing an example of a processing element of the embodiment of the present invention.

FIG. 6 is a diagram showing an example of how a control device and the processing element are connected in the embodiment of the present invention.

FIG. 7 is a diagram illustrating the structure of a command code implemented in the control device of the embodiment of the present invention.

FIG. 8 is a table showing the functions of the command codes implemented in the control device of the embodiment of the present invention.

FIG. 9 is a diagram showing an example of how the control device and the processing element are connected in the embodiment of the present invention.

FIG. 10 is a diagram showing an example of how the command of the embodiment of the present invention is stored.

FIG. 11 is a diagram showing an example of how the control device and the processing element are connected in the embodiment of the present invention.

FIG. 12 is a diagram showing an example of how the command of the embodiment of the present invention is stored.

FIG. 13 is a diagram showing a structural example of an address counter control code of the embodiment of the present invention.

FIG. 14 is a diagram showing a functional example of the address counter control code of the embodiment of the present invention.

FIG. 15 is a diagram showing a command sequence of the embodiment of the present invention.

FIG. 16 is a block diagram showing a structure where a cache device of the embodiment of the present invention is added.

FIG. 17 is a block diagram showing a structural example referred to when an operational description of the embodiment of the present invention is made.

FIG. 18 is a diagram illustrating an application implemented in the embodiment of the present invention.

FIG. 19 is a generation flow chart of program data of the embodiment of the present invention.

FIG. 20 is a diagram showing a command sequence intermediate code of the embodiment of the present invention.

FIG. 21 is a control flow chart of the process of the embodiment of the present invention.

FIG. 22 is a diagram showing command sequences assigned

to a command code memory.

FIG. 23 is a diagram of the program data assigned to a memory of the embodiment of the present invention.

FIG. 24 is a diagram showing the state of an electronic computer of the embodiment of the present invention when the power is turned on.

FIG. 25 is a timing chart illustrating the operation of the embodiment of the present invention.

FIG. 26 is a block diagram illustrating a structure where two processing devices of the embodiment of the present invention are used.

FIG. 27 is a diagram showing a command sequence assigned to the memory of the embodiment of the present invention.

FIG. 28 is a timing chart illustrating how two processing devices of the embodiment of the present invention operate.

FIG. 29 is a generation flow chart of the program data of the embodiment of the present invention.

FIG. 30 is a diagram illustrating a technique in which the whole application program is implemented in special purpose hardware in the conventional technology.

FIG. 31 is a diagram illustrating a technique in which part of application program is implemented in special purpose hardware in the conventional technology.

FIG. 32 is a diagram illustrating a technique in which part

of application program is implemented in special purpose hardware in the conventional technology.

PREFERRED EMBODIMENTS OF THE INVENTION

[0032]

Embodiments of the present invention will be described in detail with reference to the drawings. FIG. 1 is a block diagram illustrating the basic structure of an electronic computer of an embodiment of the present invention. The electronic computer 30 of the first embodiment of the present invention comprises an interface device 40 controlling the interface to the outside and transferring data, a processing device 70 performing part or all of the processing of an application program, and a control device 60 executing a command specified by the processing device 70, and is connected to an external memory device 10 via a connection network 20. The electronic computer 30 in Fig. 1 comprises two of the processing devices 70, however, it may comprise just one, three or more.

[0033]

The processing device 70 is made up of a memory unit 80 and a processor 90. FIG. 2 is a diagram showing a structural example of the processing device 70, and as constituent elements, there are a bank 101 made up of a processing element 91 and a program data memory 81, a bank 102 wherein a plurality of the selectable program data memories 81 are provided for one processing element 91, a selection memory 83, and an effective

block selection unit 82 that select one processing element from the multiple processing elements 91 and output it.

[0034]

In FIG. 2, the processing device 70 comprises two of the banks 101 and two of the banks 102, however, it suffices as long as it comprises at least one of the banks 101 or 102. Further, when there is only one bank, the effective block selection unit 82 may be omitted, and when it comprises only one bank and one program data memory 81, the selection memory 83 may be omitted.

[0035]

A program data signal S80 inputted into the processing device 70 is connected (supplied) to the program data memories 81 and the selection memory 83. The program data memory 81 is a memory that holds a program for determining the processing content of the processing element 91. The program held by the program data memory 81 is normally stored in the external memory device 10, is transferred when necessary via the connection network 20, the interface device 40, and the program data signal S80, and written into the program data memory 81. Further, when the processing element 91 is reconfigurable hardware such as an FPGA, the program held by the program data memory 81 becomes a program for creating a logic circuit of the processing element 91.

[0036]

The processing element 91 performs processing according to the program held in the program data memory 81 connected to it. This can be realized by reconfigurable hardware as shown in FIG. 3 for instance, however, it may also be realized by a CPU as shown in FIG. 4.

[0037]

The selection memory 83 is connected to an effective bank selection unit 92 and the effective block selection unit 82, and holds information for selecting a bank to be made effective in the processing device 70 and for selecting a program data memory 81 to be made effective in a bank constituted by a plurality of the program data memories 81 such as the bank 102. Since switching to a program data memory 81 is done instantly, the processing element 91 can instantly start the processing corresponding to a new program if the selected program data memory 81 has ended storing the program. However, if it has not ended storing the program, it will be necessary to wait until the data is transferred and the program is stored.

[0038]

A command signal S91 outputted from the processing device 70 is generated by the processing element 91 of the processing device 70. An interrupt signal S92 inputted into the processing device 70 is inputted into the processing element 91 of the processing device 70, and is used during the processing process. A processing data signal S93 inputted into/outputted

from the processing device 70 is connected to the processor 90, and is used as data necessary for processing and input/output line for processed data.

[0039]

The command signal S91, the interrupt signal S92 and the processing data signal S93 are inputted into/outputted from the processor 90 of the bank made effective by the selection memory 83 when the processing device 70 comprises a plurality of banks. Although all the data inputted into the processing device 70 are connected to all the processors 90 in FIG. 2, the selection memory 83 may control so as to cut some of the inputted data so that the data are not inputted into ineffective banks.

[0040]

Next, each of the constituent elements 91 will be described in detail. As shown in FIG. 3, the processing element 91 is made up of reconfigurable hardware such as FPGA and PLD. When there is a plurality of the constituent [processing] elements 91, one or multiple of them may be constituted by a CPU 120 as shown in FIG. 4. The constituent elements 91 constituted by the CPU 120 is primarily used to perform the processing of an application program by assigning part of it to be processed by a high-level language program and to perform control within the processing device 70. When the processing element 91 is constituted by the CPU 120, a program data connection line S110 in FIG. 4 may be omitted since a program can be read via the

processing data signal S93.

[0041]

As shown in FIG. 5, the control device 60 of the electronic computer 30 may be realized with reconfigurable hardware using a reconfigurable control device R60. As shown in FIG. 5, the reconfigurable control device R60 is set using a program data connection line S101, and the content of processing can be varied according to the content set. Further, the command signal S91, the interrupt signal S92, and the processing data signal S93 are inputted/outputted via the interface device 40.

[0042]

The interface device 40 connects the processing device 70 and the control device 60 to the connection network 20 external to the electronic computer 30, and outputs a command signal S41 to the control signal 60 based on a proper protocol of the connection network 20 when the communication from the connection network 20 to the electronic computer 30 about the control occurs. When the communication from the control device 60 to the interface device 40 regarding the control of an interrupt signal S42 occurs, it similarly relays the message to the specified connection target via the connection network 20 using the proper protocol.

[0043]

When an access from the inside of the electronic computer 30 to the outside occurs, the interface device 40 accesses the

outside based on the proper protocol. When an access from the outside of the electronic computer 30 to the processing device 70 occurs, the interface device 40 accesses based on the proper protocol. The control device 60 receives the command signal S41 sent from a device external to the electronic computer 30 via the interface device 40 and the command signal S91 outputted from the processing device 70, interprets, and executes the received command. An example of a protocol when the processing device 70 issues the command signal S91 to the control device 60 is shown in FIG. 6. This protocol may be applied as a protocol between the interface device 40 and the control device 60.

[0044]

FIG. 6 is a diagram showing a method in which the processing device 70 directly sends a command code signal S912 along with a request signal S911. The control device 60 receives the command code signal S912, performs processing according to the content of the command code, and returns a acknowledgement signal S921 when the processing is ended.

[0045]

An example of the command code that the control device 60 interprets and executes is shown in FIGS. 7 and 8. FIG. 7 is a diagram showing the structure of the command code, and a command code A10 is made up of a command code name A11 and a command code parameter A12. FIG. 8 is a table showing the execution contents of commands and shows six commands, which

will be explained hereinafter.

[0046]

"Activate" controls the effective bank selection unit 92 and the effective block selection unit 82 by writing a code specified by the command code parameter A12 into the selection memory 83 and connects the selected program data memory 81 to the processing element 91 in the same bank. For instance, when the processing element 91 is constituted by reconfigurable hardware as shown in FIG. 3, "activate" means setting the program data into the reconfigurable hardware, and the activated reconfigurable hardware starts processing according to the content of the program data memory 81 right away.

[0047]

"Halt" halts the operation of the processing device 70 specified by the command code parameter A12.

"Interrupt" issues a specified interrupt vector signal S922 to the processing device 70 specified by the command code parameter A12.

"Load_prg" transfers the program data stored in the external memory device 10 and other optional memory device to the region of the program data memory 81 specified by the command code parameter A12. "Cancel_prg" cancels the transfer started by "load_prg". "Wait_prg" waits until the transfer started by "load_prg" is complete.

[0048]

The control device 60 interprets any command set of the commands shown in FIG. 8, performs proper processing for each command, and outputs an interrupt signal including a acknowledgement signal to the outside of the electronic computer 30 via the processing device 70 and the interface device 40. The interpretation of the commands, processing, and interruption may be performed for each processing device 70 in parallel.

[0049]

Another example of the protocol when the processing device 70 issues the command signal S91 to the control device 60 is shown in FIGS. 9 and 11. This protocol may be applied as a protocol between the interface device 40 and the control device 60.

[0050]

FIG. 9 is a diagram showing a method in which the control device 60 comprises a command code reference device 61 and a command code memory 63, and the processing device 70 sends an address signal S913 along with the request signal S911. FIG. 10 shows an example of a command code stored in the command code memory 63. The address signal S913 specifies the address of the command code memory 63 where a command that one wants the control device 60 to execute is stored.

[0051]

The control device 60 receives the address signal S913,

refers to the command code memory 63 using the command code reference device 61, executes the command corresponding to the address signal S913, and then returns the acknowledgement signal S921 when the processing is ended.

[0052]

The command code memory 63 may be any memory such as a memory external to the control device 60 which can be referred to by the control device 60 or the external memory device 10. Further, when there is a plurality of the processing devices 70, a plurality of the command code reference devices 61 may be provided within the control device 60 so that the commands are processed in parallel, or as many the command code reference devices 61 as the processing devices 70 may be provided.

[0053]

The protocol shown in FIG. 9 requires the command code reference device 61 and the command code memory 63 in the control device 60, however, a fewer number of connection signal lines are required between the control device 60 and the processing device 70 compared with the protocol in FIG. 6 since the bit number of memory address can generally be fewer than the bit number of data.

[0054]

FIG. 11 is a diagram showing a method in which the control device 60 comprises the command code reference device 61 and its address counter 62, and the processing device 70 controls the

address counter 62 using the request signal S911 along with address operation signals S914 and S915 and sends addresses to the control device 60 using the address signal S913 when necessary. The command code reference device 61 and its address counter 62 may be provided for every processing device 70, to which they are connected.

[0055]

When the adr_ena address counter operation signal S914 is effective and the direct/offset address counter operation signal S915 indicates "direct," the address signal S913 sent from the processing device 70 is stored in the address counter 62. When the adr_ena address counter operation signal S914 is effective and the address counter operation signal S915 indicates "offset," the value of the address signal S913 sent from the processing device 70 is added to the address counter 62. When the adr_ena address counter operation signal S914 is ineffective, the address signal S913 sent from the processing device 70 is ignored and the value of the address counter 62 is held.

[0056]

When the control of the address counter 62 by the processing device 70 is complete, the control device 60 processes the command stored, referring to the command code memory 63 where the command is stored and using the value of the address counter 62 and the command code reference device 61, and returns the acknowledgement signal S921 after the

processing is complete.

[0057]

FIG. 12 is a diagram showing an example of a command code stored in the command code memory 63 in the structure shown in FIG. 11. As shown in FIG. 12, the command code memory 63 stores an address counter control code A20 and a flag A30 besides the command code A10, however, they may be excluded. When the format in FIG. 12 is employed, the control device 60 performs the processing specified by the address counter control code A20 after the processing written in the command code A10 is complete and before the acknowledgement signal S921 is returned to the processing device 70.

[0058]

FIG. 13 shows the details of the address counter control code A20. The address counter control code A20 is made up of an address counter control code name A21 and an address counter control code parameter A22, its parameter. FIG. 14 shows examples of the address counter control code. "Load_adr" sets the value of the address counter control code parameter A22 as a new value of the address counter 62. "Add_adr" adds the value of the address counter control code parameter A22 and the value of the address counter 62. "Push_adr" does not show the current value of the address counter 62, but stores it in an address counter stack and sets the value of the address counter control code parameter A22 as a new value of the address counter 62.

"Pop_adr" instruction takes out a value from the address counter stack and sets the value as a new value of the address counter 62. The address counter stack may be provided in the command code reference device 61.

[0059]

Further, the flag A30 is used as a flag indicating whether or not the command reference and execution continues to be performed using a new value of the address counter 62 after the address counter control code A20 has been executed. Hereinafter, the flag indicating that the command execution continues is called "cont," and the one indicating that the execution is stopped is called "stop."

For instance, in the structure shown in FIG. 11, when the command code memory 63 stores a command sequence shown in FIG. 15, and the processing device 70 sets the address signal S913 to 100, makes the address counter operation signal S914 ineffective, and issues a command, the control device 60 executes command codes Y100, Y101, and Y200 in order.

[0060]

Therefore, in a system where the command code reference device 61 has a built-in address counter 62 and the address counter 62 is controlled by the address counter control code A20, the processor 90 within the processing device 70 can generate the address signal S913 with low hardware resources since the processing device 70 has to output the address signal S913 only

when necessary (when a command is issued).

[0061]

As to the protocol between the processing device 70 and the control device 60, one may be selected from the ones shown in FIGS. 6, 9, and 14 or any combination of these may be used. Or a plurality of structures may be combined adding control lines as necessary so that it is possible to switch between protocols. For instance, a control line may be added so that it is possible to select between the protocol in FIG. 6 and the protocol in FIG. 11.

[0062]

Next, in order to improve the speed of the data transfer between the electronic computer 30 and the outside, a cache device 50 including a cache controller 130 may be added to the electronic computer 30 of FIG. 1, controlling the cache controller 130 with commands as shown in FIG. 16. In FIG. 16, the cache device 50 has three cache controllers 130, which are respectively connected to the control device 60, the memory unit 80, and the processor 90, however, one single cache controller 130 may be connected to all of them. A plurality of ports of cache memories 140 may be provided in the cache controller 130. Further, an address translation device 150 may be shared among a plurality of cache controllers.

[0063]

The cache controller 130 includes the cache memory 140 temporarily holding data that is stored for example in the

external memory device 10 and that the processing device accesses and the address translation device 150. The cache controller 130 is controlled by commands, transfers primarily the data between the cache memory 140 and the external memory device 10 and the data between the cache memory 140 and the control device 60 or the processing device 70, and operates in parallel with the processing device 70 and the control device 60.

[0064]

The address translation device 150 is a device that bilaterally translates addresses between the address space of the processing device 70 and the address space of the interface device 40 and is able to have an independent address space within the processing device 70. Further, it can also define an independent address space for every processing device 70 by providing an address translation device 150 for every processing device 70.

[0065]

Further, the address translation device 151 provided in the interface device 40 performs address translation on the differential of the address space between the electronic computer 30 and the external memory device 10 and other devices connected to the electronic computer 30 via the connection network 20. The processing device 70 controls the address translation device 150 by having the control device 60 execute commands.

[0066]

Commands for controlling the cache controller are not mentioned in FIG. 8, however, for instance, when the address translation device 150 comprises a translation buffer storing a pair of a translated address and translator address, commands that perform controls such as to register to the buffer, delete, and replace are needed, and also a command that sets a particular region within the cache memory 140 as a local memory region exclusively for the processing device 70 may be added.

[0067]

Further, by analyzing the data flow, performing scheduling, and describing a cache that will be used and the control of this cache as a command sequence in advance, the processing by the processing device 70 and the control of the data flow by the control device 60 can be performed in parallel, improving the processing capacity of the processing device 70. In addition, since the processing device 70 issues a command only when necessary, excessive overhead does not occur.

[0068]

For instance, it becomes possible to load data into the cache memory 140 in advance before each device accesses the cache memory 140 by controlling the cache controller 130 with commands, realizing more efficient data transfer. Further, from the above descriptions, it is apparent that, even if a plurality of the present electronic computers 30 are connected to the

connection network 20, they will be able to communicate with and control each other and the present electronic computer 30 is freely expandable. In addition, the present electronic computer 30 may be realized as an LSI comprising at least one of the present electronic computer 30 or a part of the present electronic computer 30 may be realized as an LSI. Further, the present electronic computer 30 may be logically implemented in reconfigurable hardware such as an FPGA and a PLD.

[0069]

Next, the operation of the electronic computer 30 of the embodiment of the present invention will be described with reference to the drawings. Although the various structures of the electronic computer 30 have been described, a structure shown in FIG. 17 will be used to explain the operation. First, a method for generating a program that divides an application program, the object processed by the electronic computer, into processing units stored in program data memories 811 to 813 and that generates the commands executed by the control device 60 will be explained, and then the operation and control method of the electronic computer 30 with the structure shown in FIG. 17 will be described.

[0070]

Referring to FIG. 17, a processing device 71 comprises one bank of a processing element 110 realized by reconfigurable hardware and connected to the three program data memories 811,

812, and 813 via the effective block selection 82, the control device 60 and the processing device 71 are connected by the protocol shown in FIG. 11 (minus the address counter operation signal S915 indicating "direct/offset"), and the command signal S41 and the interrupt signal S42 between the control device 60 and the interface device 40 are omitted. The command set implemented in the control device 60 in Fig. 17 is as shown in FIG. 8, and the address counter control code implemented in the control device 60 is as shown in FIGS. 13 and 14. Further, the cache device 50 is omitted.

[0071]

FIG. 18 is a control flowchart showing an example of the flow of the process content of the application executed by the processing element 110 shown in FIG. 17. As shown in FIG. 18, the application goes through an initial state C0, executes processes shown in the control flow while dynamically switching among them, and reaches a complete state C9 after all the processes are done. The initial state C0 is a state in which it is ready to start a process P1. The complete state C9 is a state in which all the functions of the processing device 71 in FIG. 17 come to a halt, for instance.

[0072]

The processing content of the application in FIG. 18 has five states (C1, C2, C3, C4, and C5) and four kinds of process contents (P1, P2, P3, and P4), and the application moves from the

initial state C0 to the state C1 unconditionally. In the state C1, it performs the process P1, and moves to the state 2 with a condition F1. In the state C2, it performs the process P2, and moves to the state 3 with a condition F2a or to the state 4 with a condition F2b. In the state C3, it performs the process P1 and moves to the state 5 with a condition F3. In the state C4, it performs the process P3 and moves to the state 5 with a condition F4. In the state C5, it performs the process P4 and moves to the complete state C9 with a condition F5.

[0073]

FIG. 19 is a flowchart showing how the program executing the application is generated. The flow shown in FIG. 19 is made up of a control flow analysis procedure M1 generating a command sequence executed after each process, a command sequence implementation procedure M2 translating the command sequence into a data string, and a program data generation procedure M3 generating program data, and the control flow of the whole application, completion, the structural information of the electronic computer 30, and its command sets are inputted into the flow outputting a command sequence code indicating the initial state C0 of the present electronic computer, all the program data used within all the processing devices, and the command sequence code to which they refer. The control flow analysis procedure M1, the command sequence implementation procedure M2, and the program data generation procedure M3 are

respectively realized by a program.

[0074]

In the control flow analysis procedure M1, all the processes (P1 to P4), the state (C1 to C5) corresponding to each process, and the transition condition (F1 to F5) and target corresponding to each state are analyzed. And the intermediate code of a command sequence where it is possible to move to a next state by assigning each process to one of the program data memories 811 to 813, switching the effective block selection unit 82, and continuously executing each process is generated.

[0075]

An example of the intermediate code of the command sequence resulted from the analysis of the application in the control flow analysis procedure M1 in FIG. 18 is shown in FIG. 20. Here, when the power is turned on, the effective block selection unit 82 selects the program data memory 811. In FIG. 20, 812, a parameter of a command "load_prg 812, PM1" of a command sequence SQ0 indicates that the program data generated to execute the process P1 in the process element 110 is loaded into the program data memory 812, and PM1 indicates a region of the memory where this program data is stored. The program data PM1 specified by PM1 may be stored in any memory including the external memory device 10. At this point, the program data PM1 itself has not been generated, therefore it points to an empty region. Program data PM2, PM3, and PM4

respectively specify regions of the memory where program data generated to execute the process P2, the process P3, and the process P4 are stored.

[0076]

In FIG. 20, for instance, the C4 is a state in which the process P3 is being executed, and when the condition F4 is met in this state, a command sequence SQ4 starts. The process content of the command sequence SQ4 does not start until the transfer of the program data PM4 to the program data memory 813 by "wait_prg 813, PM4" is complete and does start when the program data memory 813 is selected by "activate". When the program data memory 813 is activated, the processing element 110 starts the processing determined by the program held in the program data memory 813. At this point, since the program data executing the process PM4 is being held in the program data memory 813, the processing element 110 starts the process PM4. In other words, this means a move to the state C5.

[0077]

As described, by adding execution procedures of the command sequence to the process content, it is possible to move to a next state. Furthermore, since the commands are issued at the same timing as when the processing device (the processing element) itself detects a predetermined condition, the timing of transition can be included in the process content, improving process efficiency. Further, in FIG. 20, since it is necessary to

recognize whether the current state is C1 or C3 when the process P1 is being performed, before the program data memory that performs the process P1 is activated, an interrupt vector is set in the processing device 71 that performs the process P1, and transition conditions are revised to ones using its value. For instance, the setting of the interrupt vector of the state C1 is executed by "interrupt 71, C1" of the SQ0.

[0078]

How the command sequences in FIG. 20 are assigned to memories in the command sequence implementation procedure M2 is shown in FIG. 22, and the control flow of each process translating the call of the command sequence in each process into the protocol of the interface between the control device 60 and the processing device 71 in the structure shown in FIG. 17 is shown in FIG. 21.

[0079]

FIGS. 21 and 22 include a command sequence SQ0A executed first after the power is turned on. By executing the command sequence SQ0A, the application moves to the initial state C0. In the state C0, the program data memory 811 is selected by the effective block selection unit 82, therefore the processing element 110 starts the operation of the program stored in the program data memory 811 after the power is turned on.

[0080]

As shown in FIG. 21, each process becomes a control flow

where each command sequence issuing process is added to the state when the power is turned on and each of the original process (P1, P2, P3, and P4). For instance, in the process P2 in FIG. 21, a command issuing process SQ2aA is executed when the condition F2a is met, and a command issuing process SQ2bA is executed when the condition F2b is met. When the processing device 71 executes the command issuing process SQ2aA, the request signal S911, the adr_ena address counter operation signal S914, and the address signal S913 indicating ADR002 are outputted. As shown in FIG. 22, the address ADR002 specifies a command sequence SQ2a, therefore the control device 60 execute the content of the SQ2a.

[0081]

Further, as shown in FIG. 20, since the operations of the command sequences SQ3 and SQ4 are identical, they can share one base address value, ADR004 as indicated in FIG. 22. In addition, as shown in FIG. 18, since the state C5 follows the state C3 or the state C4, "add_adr" is used as the address counter control code of the offset value +1 of the address ADR004 specifying the command sequences SQ3 and SQ4 instead of "load_adr" to set the next value of the address counter 62 (ADR004 offset + 2) in FIG. 22. With such a design, the processing device 71 does not have to output extra addresses.

[0082]

Finally, the program data executing each process shown in

FIG. 21 is generated in the program data generation procedure M3. FIG. 23 shows an example where each program data generated is stored in a memory. The memory region where the generated program data PM1, PM2, PM3 and PM4 are stored is reflected in the parameter of each command sequence in FIG. 22.

[0083]

Further, by having the electronic computer comprise the address translation device as shown in FIG. 16, the address specifying the memory where the memory addresses and commands that each process accesses are stored can be designed in an independent address space. Regarding program data PM0, since it needs to be executed right after power-on of the electronic computer with the structure shown in FIG. 17, this program data PM0 must be stored in the program data memory 811 in advance. Therefore, the initial state of the electronic computer at power-on in the present embodiment is as shown in the list of FIG. 24.

[0084]

Next, the operation and control method of the electronic computer 30 will be described. FIG. 25 is a timing chart illustrating the operations described above. Explanations will be made using FIGS. 25, 17, 18, 21, 22, 23, and 24.

[0085]

The horizontal rows of FIG. 25 show the values of the address counter 62 in the control device 60, the operational

contents of the control device 60, the operational contents of the processing element 110, the contents of the interrupt vector signal S922 inputted into the processing element 110, the contents of the program data respectively held by the program data memories 811, 812, and 813, and the status of the transfer of the program data, and the vertical column shows the lapse of time in descending order from T101. The parts shown in mesh in the columns of the program data memories 811, 812, and 813 indicate that they are being activated.

[0086]

At the timing of T101, the electronic computer 30 is in the state when the power is turned on as shown in FIG. 24, and the program data PM0 that performs the command issuing process SQ0A is stored in the program data memory 811.

[0087]

At T102, the processing element 110 starts the operation of the PM0 and executes the command issuing process SQ0A. The command reference device 61 of the control device 60 sets the address counter 62 to the address value ADR000 when it receives the command issuing process SQ0A, and reads the commands stored at ADR000 of the command code memory 63 where the command sequences of FIG. 22 are stored, and executes "load_prg" at T103. At T103, the control device 60 executes "load_prg" and starts transferring the program data PM1 to the program data memory 812. The information as to where the

program data PM1 is stored is included in PM1, a parameter of "load_prg," and here it is assumed that it is stored in the external memory device 10 as shown in FIG. 23, however, it may be stored in any memory other than the external memory device10.

[0088]

Referring to FIG. 22, since the address counter control code at ADR000 is set so as to execute a next command, the control device 60 adds one to the address counter 62 and executes the next instruction at T104. At T104, the control device 60 waits the program data PM1 to be transferred to the program data memory 812 completely, and when the transfer is ended at T105, the control device 60 continues to execute the next instruction at T106.

[0089]

Executions continue similarly, and when the control device 60 executes "activate 812" at T107, the selection memory 83 is updated to the information that selects the program data memory 812, the effective block selection unit 82 switches to the program data memory 812 following the instruction of the selection memory 83, and the processing element 110 switches its operation to start processing PM1 stored in the program data memory 812. At T108, the control device 60 ends processing a series of the command sequence started at T102 with the address counter control code "add_adr 0/stop."

[0090]

Executions continue similarly, and at T110 when the processing element 110 detects the condition F1 while executing the process P1 and confirms that the interrupt vector signal S922 is in C1, it executes a command issuing process SQ1A. Receiving the command issuing process SQ1A at T110, the control device 60 updates the value of the address counter 62 to the value ADR001.

[0091]

At T112, the control device 60 reads the command code memory 63, executes "activate 813," and switches the process content of the processing element 110 to a processing PM2 stored in the program data memory 813.

[0092]

Executions continue similarly, and at T123 when the processing element 110 detects the condition F5 while executing the process P4, it executes a command issuing process SQ5A. The control device 60, which has received the command issuing process SQ5A at T123, refers to the command code memory 63, and executes "halt" to the processing device 71 at 124. Receiving the instruction "halt," the processing device 71 halts operating and the control device 60 that has executed "halt" ends interpreting and executing the command sequence.

[0093]

As described, five pieces of program data PM0, PM1, PM2, PM3, and PM4 are generated, however, only three program data

memories 811, 812, 813 are used. This fact indicates that an application beyond the hardware resources can be implemented in the present electronic computer 30.

[0094]

The processing device 71 comprises only one bank in FIG. 17, however, when it comprises two or more banks, the program data PM0 to PM4 may be stored in the program data memories of multiple banks and executed by the processing elements made up of respective reconfigurable hardware. In this case, an output of the processing device becomes a command and output data of the processing element selected by the effective bank selection unit 92, the processing element selected by the effective bank selection unit 92 operates effectively, and processing is proceeded while switching between the program data memories and the processing elements.

[0095]

Next, the operation when there is a plurality of the processing devices 70 will be described with a structure in FIG. 26 as an example. In FIG. 26, a processing device 72 is added to the structure shown in FIG. 17 where there is one processing device 71, resulting in the structure with two processing devices 70. Further, the processing device 72 executes issuing of the command SQ0A, which triggers the operation, instead of the processing device 71. Further, the command sequence SQ5 of the processing device 71 shown in FIG. 22 is changed to a command

sequence SQ5B shown in FIG. 27, adding the execution of "interrupt 72, END71" after "halt," therefore the fact that the processing device 71 ends its operation is notified by interrupting the processing device 72.

[0096]

FIG. 28 is a timing chart illustrating the above-described operation where the processing device 72 executes the issuing of the command SQ0A and the processing device 71 notifies that it ends its operation to the processing device 72 with "interrupt."

[0097]

Referring to FIG. 28, at T201, the processing device 72 executes the command issuing process SQ0A for initializing the processing device 71 during the process P5. Receiving the command issuing process SQ0A at T201, the control device 60 executes the command sequence SQ0. The processing device 71, which has been initialized at T202, performs a series of the operation shown in FIG. 25 and executes the command issuing process SQ5A at T203. At T203, the control device 60 executes the command sequence SQ5B, halts the processing device 71 by executing "halt" at T204, and outputs the interrupt to the processing device 72 by executing "interrupt" at T205.

[0098]

After executing the command issuing process SQ0A at T201 while executing the process P5, the processing device 72 can instantly execute another process P6. And it waits for an

interrupt from the processing device 71 during the process P6 and can move to the following process P7. The processing device 72 may simply wait for the interrupt from the processing device 71 without executing the process P6.

[0099]

Further, in the structure in FIG. 26, the end state after "halt" is executed for the processing device 71 can become an initial state of the processing device 71, and the same process can be executed numerous times from the processing device 72. In addition, the processing element 72 can be structured as shown in FIGS. 2, 3, 4 and 5, and especially when the CPU 120 is a constituent element of the processing element 72, the processing element 71 may be used as a function call or system call within its software program or a thread and object including these, or an extension of the instruction of the CPU 120.

[0100]

As described, since the processing devices are synchronized utilizing the "interrupt" command when a plurality of the processing devices operate in parallel, the context of the whole process can be executed in order while improving the processing capacity.

[0101]

Further, since the electronic computer 30 can receive commands from the outside via the connection network 20 as shown in FIG. 1, the command issuing process SQ0A may be

executed externally from the electronic computer 30. Also, since the processing element of the processing device 70 can be realized with the electronic computer 30 as shown in FIG. 5, the processes may be hierarchized and subdivided when being implemented.

[0102]

In the above description of the operation, an embodiment where the structure of the present electronic computer is fixed is shown, however, for instance, when the control device 60 is the reconfigurable control device 60 as shown in FIG. 5, only commands used in the command sequence intermediate code are used as a command set in the generation flow such as the one shown in FIG. 29, and the control device 60 in which only the command subset of the command set is implemented may be employed. By having such a structure, the control device 60 can be simplified.

[0103]

The cache device 50 shown in FIG. 16 was omitted in the above description, however, it is evident that the cache device 50 can be inserted in the structures shown in FIGS. 17 and 26, the data transfer time between the electronic computer 30 and the outside can be shortened by inserting the cache device 50, and it becomes possible to have the processing device have its own address space by having the address translation device 150.

[0104]

Further, all the constituent elements including the control device 60 and the processing device 70 may be newly designed and manufactured based on the analytical results by the implementation flow.

[0105]

It is to be understood that the present invention is not limited to the above-described embodiments and each embodiment can be suitably modified within the scope of the technological idea of the present invention.

INDUSTRIAL APPLICABILITY

[0106]

As described above, according to the present invention, since even a large application program can be executed by small reconfigurable hardware by dividing the process of the application program into a plurality of processing units executed by the reconfigurable hardware while switching between programs creating a logic circuit for every processing unit, the application program can be executed at high speed and low cost, and further, since the program of the processing unit can be easily applied by changing the command sequence of the processing unit, the application of a new application program and the creation of a new electronic computer can be realized at low cost.

[0107]

Further, by having a plurality of the program data

memories holding the programs creating the logic circuits and reading the program of the next processing unit into another program data memory during the execution of a processing unit, the read time of the programs when being switched can be shortened, resulting in a shorter switching time and the improvement of the processing speed.